Centers for Disease Control and Prevention
National Immunization Program

Programmer's Guide To
The Automated Immunization Evaluation Process

Version 2.0

1.0 Introduction

The heart of an Immunization Information System is its ability to evaluate individual immunization histories for completeness and up-to-dateness. This functionality is used, first, to identify individuals who are in need of further immunization services; second, as a guide to the administration of immunizations in providers' offices; and, third, to assess the progress of providers and immunization programs, in general, toward the goal of complete immunizations for all.

The design and implementation of immunization evaluation algorithms, heretofore, has occurred in the context of individual system development. Consequently, there is considerable variation in the scope and operation of the mechanisms -- some simply count the number of doses received at specified age thresholds; others attempt to emulate standard immunization schedules exactly, by taking into account the recommended intervals between doses and other factors. In some systems, the algorithm is embedded completely in program source code or rule-based procedure, while, in others, a parameterized approach has been taken, placing the variable aspects of the process in data tables, to minimize the need to change program code when changes occur in the recommendations themselves.

There is, at present, no definitive guidance for developers on the creation of immunization evaluation algorithms; nor does there exist a mechanism for evaluating them to certify that they are correct, that is, that they give the right answers in all situations. This paper represents an attempt to provide that guidance.

First, a process definition will be offered and the design objectives and set of features of an ideal algorithm will be described. Then, an analysis of functional requirements will be made, followed by a description of the algorithm itself, including relevant parameters and process. Finally, a discussion will be made of methods for implementing the algorithm and evaluating its performance.

2.0 Immunization Evaluation Process Definition

An immunization evaluation process, or algorithm, is a function that takes, as parameters, (1) an individual history of immunizations and contraindications to immunization, (2) the individual's birthdate, (3) a set of rules, governing immunization requirements and (4) a date as of which the rules are to be applied, and returns a list of zero, one or more recommended immunizations.

An individual immunization history is a table of immunizations received by the individual; each immunization is characterized by a vaccine type and a date as of which the immunization was administered. An individual history of contraindications is a table of conditions, pertaining to an individual, which would preclude the recommendation of specified immunizations; each contraindication is characterized by a vaccine type and an expiration date, after which the contraindication would no longer be in effect.

3.0 Design Objectives

There are two major objectives for the design of an immunization evaluation algorithm:

1. It should be able to apply exactly the rules for immunization embodied in the ACIP or AAP schedules. Operationally, this means that the algorithm should make the same recommendations about specific vaccines that an expert clinician would make, when applying either of the schedules to a given immunization history.

2. The algorithm should be structured so that any changes to the ACIP or AAP schedules may be implemented without changing program source code.

These objectives are stringent ones. They represent ideal performance goals that may be approached over time through an iterative development process.

4.0 Features

An ideal immunization evaluation algorithm will have the following features:

- Any number of immunization schedules (rule sets) may be defined. The two main ones are embodied in the ACIP and AAP recommendations, but it may be desirable to define variations of them, customized for different providers, or completely different schedules for special purposes, like epidemic control. Applications using the algorithm should, when appropriate, allow users to select a desired schedule for use in specific situations.

- Within a given schedule, any number of vaccine series may be defined. A series definition will contain the recommendations for a given type of immunization.

- Any of the details of a schedule definition may be modified or deleted through an interactive process by authorized users.

- The algorithm should use the Anniversary Method for adding or subtracting time intervals to or from dates. (For an explanation of the Anniversary Method, see Section 8.0)

- The algorithm should recommend combination vaccines in preference to single-antigen vaccines, when appropriate. Combination vaccines contain immunizing agents for more than one type of immunization.

- At the option of the user or application, the evaluation function should (1) return a list of all vaccines recommended as of the specified date, or (2) the next recommended vaccine from the list, in round-robin order.

- In the event that the individual is up-to-date or complete on all defined series, the function should have the capability of projecting the earliest date at which immunizations would be recommended again.

- The function should have the capability of providing a schedule of immunization visits that an individual would have to make in order to complete all the recommended series. The schedule could be adjusted to fit any point in the individual's immunization history. In addition, this function should be capable of providing customized schedules, based on user preferences for, say, minimizing visits or injections per visit.

5.0 Functional Analysis

The analysis of functional requirements will be carried out in relation to the ACIP Immunization Schedule -- it is extremely complex, so it will be assumed that a parametric process that adequately models its recommendations will also be able to cover any other type of schedule satisfactorily.

Immunizing agents are designed to confer immunity against specific disease antigens or toxins, like measles, polio and diphtheria. One or more doses of an immunizing agent, administered over a period of time, may be required to produce long-lasting immunity.

The ACIP Immunization Schedule describes several categories, which in general correspond to individual antigens, and sets forth, for each one, the number of immunization doses that will be required to produce complete immunity, based on immunological research and clinical trials. These categories will be referred to as immunization series.

Although immunization series in the ACIP Schedule generally correspond to single antigens, this is not always so: DTP (diphtheria-tetanus-pertussis) and MMR (measles-mumps-rubella) are regarded as series, even though they each refer to multiple antigens. It would be more to the point to say that immunization series correspond to vaccines, rather than antigens, since both DTP and MMR are vaccine products. From the perspective of an immunization evaluation process, this is, in fact, a more useful approach, since immunization histories (the things being evaluated) are defined in terms of vaccines administered to individuals.

It is important to note, however, that a given series may relate to more than one vaccine (the polio series includes OPV and IPV) and a given vaccine may relate to more than one series (DTP/Hib vaccine applies both to the DTP series and the Hib series). An immunization series, then, refers to one or more vaccines, non-exclusively. (Note that, in some discussions, immunization series are referred to as vaccine groups or "families", but the structure is the same.)

As noted previously, the ACIP Schedule defines, for each series, a number of doses required to produce immunity against the respective diseases. If an individual receives the required number of doses (subject to conditions described below), he is said to be complete for that series; otherwise, he is said to be incomplete. A series, however, may be open-ended: there is, for example, no maximum number of doses for the Td (Tetanus-diphtheria) series; individuals should receive periodic doses of this vaccine throughout life.

If an individual is incomplete for a given series, he may or may not be eligible to receive the next dose at a given point in time. The ACIP Schedule prescribes minimum intervals between successive doses in a series. If an individual receives a subsequent dose of vaccine before the minimum interval of time has passed since the previous one, that dose is not counted toward the number required for the series. (Some ACIP analysts maintain that the minimum time interval for counting the next dose is different from the minimum interval for administering the next dose. Accordingly, in this guide, separate minimum interval parameters will be maintained, one for counting and the other for administering a subsequent dose, although it would be valid for both of them to have the same value in specific instances.)

When the minimum interval for administration has passed, the individual may receive the next dose in the series, but, in general, the ACIP Schedule does not recommend immunization at that point. Instead, it prescribes a recommended interval of time before the next dose is given. Until this interval has passed, the individual is said to be up-to-date for the given series.

Another way in which the ACIP Schedule represents intervals between doses in a series is by minimum age. For individuals who start each series on time, the minimum age parameters are implied by a cumulation of the minimum intervals. However, for those who start the series later, the recommended intervals may be shortened, so it is necessary to use both parameters in evaluating immunization histories.

Current formulations of the ACIP recommendations, it should be noted, allow a degree of flexibility in the specification of recommended ages for immunization. For selected series and doses, a range of ages is specified, within which immunization may be recommended. One way to implement this feature would be to allow providers to select recommended ages within this range for use in the evaluation process.

Some vaccines may not be given to individuals above a certain age. For example, oral polio vaccine is not recommended after the 18th birthday. Therefore, it must be possible to specify a maximum age for given vaccines. This parameter is not dose-specific; it applies to any doses of the specified vaccine.

Under certain conditions, the ACIP Schedule recommends accelerating or abbreviating immunization series. For individuals who start a series much later than the recommended time, it may be recommended that subsequent doses be given at the minimum, rather than recommended, intervals. Also, in these cases, the number of doses required may be reduced. These rules are dependent in some instances on the age at which the first dose in the series was received, and, in others, on the age at which the last dose was received.

Although the ACIP Schedule does not specify it, there is another kind of interval that, as a practical matter, is important to take into account: immunization history evaluations are often used to trigger follow-up activities designed to return the individual to a provider to continue the immunization process. If the recommended intervals were used for this purpose, follow-up might be performed on individuals who were already planning to return for scheduled immunizations; effort and resources could, therefore, be wasted. Most providers prefer to wait for a period of time, beyond the recommended intervals, before initiating follow-up actions. Hence, an overdue interval, should be specified, not as an evaluation parameter, but as a provider-controlled adjustment to the recommended interval, to be used in immunization recall processes.

In this discussion, it is important to note that the interval and age parameters between doses in a series are not constant: the minimum or recommended interval between dose one and dose two may be different from that between dose three and dose four. Intervals and minimum ages, therefore, must be regarded as dose-specific parameters.

Also, it should be pointed out, that, within a given series, the interval and age parameters may vary by vaccine type. So it must be possible to specify the parameters individually for each vaccine when necessary.

In some cases, the next dose in a series may not be given within a minimum interval after the administration of a vaccine that is not in the series. For example, measles vaccine may not be given within a specified interval following a rubella or mumps vaccination, and vice versa. Therefore, it must be possible to include a vaccine in a series definition for the purpose of evaluating the interval only; such vaccines would not count toward the required doses in the series, nor would they be recommended for administration.

In general, if it is determined that the next dose in a series is due, any of the eligible vaccines in the series may be administered. However, combination vaccines (those that apply to more than one series) should be used in preference to single-series vaccines, if all the series they apply to are currently recommended; otherwise, the single-series vaccines should be used.

Also, in applications that have inventory control modules, it would be useful for the evaluation algorithm to give preference to vaccine types that are available in stock.

Beyond this, if more than one vaccine is eligible for use at a given point in time, it would be useful to specify an order of preference in the series definition process; otherwise, the automated selection of a vaccine type would be arbitrary.

The foregoing analysis shows the ACIP Immunization Schedule to be a very complex set of rules. In order to design a general algorithm to model this process, that does not have to be changed over time, an equally complex set of parameters must be specified. The next sections will attempt to describe an algorithm and parameter set that can achieve this objective.

6.0 Data Structure

6.1 Evaluation Parameters

The parameters that make up the evaluation rule set may be conceptualized

as four linked data tables with associated parameters:

1. Schedule Definitions

Schedule Name*
Age at First Dose (any series)

2. Series Definitions

Schedule Name*
Series Name*
Age at First Dose in Series*
Number of Doses Required for Series
Unlimited Doses Flag

Series Return Date**

3. Series Vaccine Definitions

Schedule Name*
Series Name*
Age at First Dose in Series*
Series Vaccine Name*
Interval Only Flag
Vaccine Sequence Number
Minimum Age
Maximum Age

Vaccine Eligible Flag**
Eligible Vaccine Count**
Vaccine Combination Count**

4. Vaccine Dose Definitions

Schedule Name*
Series Name*
Age at First Dose in Series*
Series Vaccine Name*
Dose Number*
Minimum Interval to Count Dose
Minimum Interval to Administer Dose
Recommended Interval
Minimum Age
Recommended Age
Valid Recommended Age Range (From and To Parameters)
Skip Age (age at which next dose may be skipped)

* columns which, concatenated, form a unique key for table
** algorithm working storage, not stored with parameters

The tables, linked by their common fields, form a hierarchical dataset. Each Schedule Definition would be represented, with its associated attributes, in the Schedule Definition table; it would have one or more Series Definitions in the Series Definition table, and each series would have one or more Vaccine Definitions in the Series Vaccine Definitions table, and so on. This is an

abstract description of the data structure. Later on, an actual implementation will be described.

The evaluation algorithm will use these parameters to make judgments about an individual's immunization needs. In addition to the evaluation rule set, the following data elements will be used from the Immunization and Contraindication Histories:

6.2 Immunization Data

Immunization History

Age at First Immunization (any series)
Age at First Immunization of a Specified Series
Age at Last Immunization of a Specified Series
Dose Number of Last Dose Received in a Specified Series
List of Immunizations and Dates Received, in Date Order

Contraindications History

List of Contraindications - vaccine type and expiration date

Finally, from the application, or user of the algorithm, two parameters must be specified at the time an evaluation is performed: Evaluation Mode and Evaluation Date.

7.0 Algorithm

The basic algorithm evaluates one immunization series in the selected schedule. It may be represented as a function:

Recommendation = GetRecommendation(Series, Vaccine, Mode, Date);

Series is specified as an input parameter; it is referred to in the function as the SeriesBeingEvaluated. The return parameter, Recommendation, will have one of the following values: COMPLETE, UP_TO_DATE, NO_RECOMMEND, CONTRAINDICATED, WRONG_ALTERNATE or RECOMMEND.

If the return value is RECOMMEND, the parameter, Vaccine, will contain the name of the vaccine that is recommended.

Mode is specified as an input parameter, with one of the following values: RECOMMENDED, OVERDUE or MINIMUM. This determines which set of dose interval parameters are used in the evaluation.

Date is specified as an input parameter and represents the date as of which the evaluation is to be made. It may be any date, on or after the birthdate of the individual. All interval and age determinations will be computed relative to this date. The default is the System Date.

If the value returned by the function is UP_TO_DATE, Date will contain the earliest date at which immunizations will be recommended again. This date will be maintained in the function in a data element, SeriesReturnDate.

GetRecommendation has the following structure:

```
{
  SetDefaultMode();
  Recommendation = GetEligibleVaccines( Series );
```

```
  if ( Recommendation == RECOMMEND )
  {
    CheckOtherSeriesForEligibleVaccines();
    Vaccine = SelectEligibleVaccineForRecommendation();
  }
  if ( Recommendation == UP_TO_DATE )
    Date = SeriesReturnDate;
  return Recommendation;
}
```

The function, SetDefaultMode, does the following: if Mode is OVERDUE, the function returns with no change in the value of the Mode parameter. Otherwise, if the individual's age at which the first dose of any series was received is equal to, or greater than, the AgeAtFirstDoseAnySeries parameter, then Mode is set to MINIMUM, else it is set to RECOMMENDED.

The function, GetEligibleVaccines, has the following structure:

```
{
  DoseCount = GetSeriesDoseCountFromHistory();
  RightAlternate = CheckForRightAlternateSeries();
  if ( RightAlternate == FALSE )
    return Recommendation = WRONG_ALTERNATE;
  if ( DoseCount >= DosesRequiredForSeries && !UnlimitedFlag )
    return Recommendation = COMPLETE;
  SetAllSeriesVaccinesToEligible();
  ExcludeContraindicatedVaccines();
  if ( EligibleVaccineCount == 0 )
    return Recommendation = CONTRAINDICATED;
  ExcludeVaccinesOutsideAgeRangeForSeries();
  if ( EligibleVaccineCount == 0 )
    return Recommendation = NO_RECOMMEND;
  ExcludeVaccinesNotEligibleForNextDose();
  if ( EligibleVaccineCount == 0 )
    return Recommendation = UP_TO_DATE;
  else
    return Recommendation = RECOMMEND;
}
```

The function, GetSeriesDoseCountFromHistory, does the following: it scans the individual's list of immunizations in date order and, for each vaccine type that is included in the list of vaccines defined in the series parameters, a dose count is incremented and the immunization is copied to a list of immunizations received for the series. In other words, the function pulls out from the complete history just those immunizations that are included in the series definition and counts them. The function then scans the new list of series immunizations and decrements the dose count for any that were given within the minimum interval defined for the previous dose, but leaves the immunization(s) in the series list. Finally, the ages of the individual when the first and last series doses were given are computed.

The function, CheckForRightAlternateSeries, scans all alternate parameter definitions for the same series, if any, to determine if the current series is the right one to use in the evaluation. Alternate Series may be defined with different parameters for the age at which the first series dose was given. The function returns TRUE if the age at which the first series doses was received (computed in GetSeriesDoseCountFromHistory) is greater than or equal to the current Series Age At First Dose parameter, but less than the same parameter in the next alternate series definition, taken in order of the parameter. Otherwise, it returns FALSE. An example may be helpful here: for the Hib (Haemophilus Influenzae b) immunization, the ACIP Schedule defines four independent series, depending on the age at which the first dose in the series is received: if the series is started at two months of age, four doses are required at appropriate intervals; if it is started at seven months of age, three doses are required; at 15 months,

two doses are required and at 59 months only one dose should be given. Four parameter series (one regular and three alternates should therefore be defined. For an individual starting the series at 13 months of age, the function should select the alternate keyed to the seven-month starting age and, hence, the algorithm should call for three doses to be given.

If CheckForRightAlternateSeries returns FALSE, then the subsequent functions in GetEligibleVaccines are not evaluated and GetEligibleVaccines returns WRONG_ALTERNATE.

If CheckForRightAlternateSeries returns TRUE, then the SeriesBeingEvaluated is the appropriate one. The count of series doses received by the individual is then compared to the number of doses required in the Series Definition. If the number of doses received is equal to or greater than the number required, then GetEligibleVaccines returns COMPLETE.

The function, SetAllSeriesVaccinesToEligible, initializes the VaccineEligibleFlag to TRUE for each vaccine in the Series Vaccine Definition and sets EligibleVaccineCount equal to the number of vaccines.

The function, ExcludeContraindicatedVaccines, checks each vaccine in the Series Vaccine Definition to see if it appears in the list of current contraindications. Any series vaccine found in the contraindication history with an unexpired date is flagged as not eligible for recommendation (VaccineEligibleFlag is set to FALSE and EligibleVaccineCount is decremented).

For any vaccine flagged as ineligible because of contraindication, the date when the contraindication would expire is compared to the current value of SeriesReturnDate. If the expiration date is earlier, then SeriesReturnDate is set equal to it.

If, on return from this function, EligibleVaccineCount is zero, GetEligibleVaccines returns CONTRAINDICATED.

The function, ExcludeVaccinesOutsideAgeRangeForSeries, checks the minimum and maximum age parameters for each vaccine in the series. If the individual's age as of the evaluation date falls outside this range for a given vaccine, the vaccine is flagged as not eligible for recommendation (VaccineEligibleFlag is set to FALSE and EligibleVaccineCount is decremented).

For any vaccine flagged as ineligible because the individual is younger than the minimum age, the date when the vaccine would be eligible is computed and compared to SeriesReturnDate. If it is earlier, SeriesReturnDate is set equal to it.

If, on return from this function, EligibleVaccineCount is zero, GetEligibleVaccines returns NO_RECOMMEND.

The function, ExcludeVaccinesNotEligibleForNextDose, examines the dose interval and age parameters for each eligible vaccine in the series and compares them to the actual intervals and age of the individual at the last series dose that was received. Specifically, the dose number of the last dose received is determined from the list of series doses (compiled in GetSeriesDoseCountFromHistory) and this number is used to reference the appropriate dose-specific parameters in each Vaccine Dose Definition. Next, the Skip Age parameter for the specified dose number is evaluated: if the individual's age at that time is equal to or greater than the Skip Age (and Skip Age is not 0) then the last dose number is incremented and the remaining dose-specific parameters are evaluated, using this number. The effect will be to skip the next dose in the series. Then, the dose-specific age and interval parameters are evaluated as follows: for each eligible vaccine, if the appropriate interval has not passed since the last dose was administered, or if the individual is younger than the minimum age for the next dose, the vaccine is marked ineligible for recommendation (VaccineEligibleFlag is set to FALSE and EligibleVaccineCount is decremented). Note that any vaccines, included in the Series Vaccine Definition only for interval evaluation (IntervalOnlyFlag is TRUE), are also marked ineligible at this point.

For any vaccine marked ineligible, the date when it would be recommended is computed and, if earlier than the current value of SeriesReturnDate, SeriesReturnDate is set equal to it.

The particular interval parameter (Minimum, Recommended or Overdue), selected for evaluation, is determined by the Mode parameter that is set when GetRecommendation is called. The default is Recommended.

If, on return from this function, EligibleVaccineCount is zero, GetEligibleVaccines returns UP_TO_DATE; otherwise, it returns RECOMMEND.

If GetEligibleVaccines returns a value other than RECOMMEND, then GetRecommendation returns that value. On the other hand, if GetEligibleVaccines returns RECOMMEND, then these functions are executed:

```
{
  CheckOtherSeriesForEligibleVaccines();
  Vaccine = SelectEligibleVaccineForRecommendation();
}
```

The function, CheckOtherSeriesForEligibleVaccines(), has the following structure:

```
{
  For ( each series except the SeriesBeingEvaluated )
  {
    OtherRecommendation = GetEligibleVaccines( OtherSeriesName );
    If ( OtherRecommendation == WRONG_ALTERNATE;
      continue;
    If ( OtherRecommendation == RECOMMEND )
      IncrementCombinationCountsInSeriesBeingEvaluated();
    else
      MakeCombinationsIneligibleInSeriesBeingEvaluated();
  }
}
```

This function checks all other series in the Schedule, except the SeriesBeingEvaluated, to determine which vaccines in them are eligible for recommendation. Any vaccines which are found to be eligible in both the SeriesBeingEvaluated and in other series will be given preference by incrementing the parameter, VaccineCombinationCount; conversely, any vaccines in the SeriesBeingEvaluated which are COMPLETE or UP_TO_DATE in other series, are marked ineligible (VaccineEligibleFlag set to FALSE and EligibleVaccineCount decremented).

The function, SelectEligibleVaccineForRecommendation, looks at each eligible vaccine in the SeriesBeingEvaluated and selects the one(s) with the highest VaccineCombinationCount. If more than one vaccine is selected, the one in which VaccineSequenceNumber has the lowest value is returned.

GetRecommendation, now completely described, evaluates one series in a schedule. As such, it will not usually be executed directly by an application, but instead will be embedded in another function, which will evaluate the entire schedule. This function would execute GetRecommendation for each series in the schedule and would return, depending on an option specified by the calling application, either (1) a list of all recommended immunizations, (2) the next recommendation in the schedule or (3) if the individual is up to date on all series, the earliest date at which the next immunization(s) would be recommended, along with a list of them.

8.0 Anniversary Method for Adding Intervals to Dates

The algorithm for evaluating immunization histories involves adding and subtracting time intervals to and from dates. Most modern programming languages have functions for performing date "arithmetic", which allow time intervals, expressed in days, to be added to or subtracted from dates. For this reason, most evaluation algorithms convert all time intervals to days for processing. Immunization recommendations, however, are typically stated in terms of months and years and these terms can not be unambiguously converted into days: a month may be 28 to

31 days long and a year may be 365 or 366 days long. It is, therefore, not possible to evaluate immunization intervals accurately, using what may be called the Conversion Method.

The CDC Model Immunization Information System uses another method, called the Anniversary Method for combining dates and time intervals, which emulates the common sense method for dealing with months and years: most people would say that a child is one year old on his birthday in the year following birth; similarly, he would be one month old on his birthday in the month following birth. Two months from the last immunization would be the same month-day, two months hence. Procedurally, the "anniversary method" calls for adding years to dates by incrementing the date-year while holding the month and day constant; months are added by incrementing the date-month (and date-year, if necessary) while holding the day constant.

In this approach, time intervals are expressed in text form as numbers, followed by modifiers to indicate the units involved, for example, "6 years" or "3 months". A robust algorithm would allow multiple terms to be combined in one expression, like "2 years 3 months". The terms could appear in the expression in any order. Weeks and days could also be evaluated and variant forms of the modifiers ("yr", "mo", "wk", "dy", "y", "m", "d") could be used. The following, therefore, would be a valid interval expression: "4 y 3 m 2 w 1 d".

To provide a convenient semantic for the application of the Anniversary Method, the following functions are defined:

```
Date AddIntervalToDate( Date, Interval );

Date SubtractIntervalFromDate( Date, Interval );

BOOL IsSecondDateLaterThanFirstPlusInterval( Date, Date, Interval );
```

The first two functions return a date which is the result of adding (or subtracting) the interval to the specified input date. The third function returns TRUE if the second date is later than the date that is the result of adding the first date and the interval together; otherwise, FALSE is returned.

A number of modern programming languages, like Visual Basic, provide functions similar to these.

Examples

A child, born on 10/12/1995, will be one year old on his birthday in 1996. This one-year-old date is calculated first by the Conversion Method with an incorrect result and then by the Anniversary method with a correct result:

WRONG
```
int nYear = 365;
CString strBirthDate = "10/12/1995";
CString strOneYearOldDate = AddDaysToDate( strBirthDate, nYear );
ASSERT( strOneYearOldDate == "10/11/1996" );
```

RIGHT
```
CString strInterval = "1 year";
CString strBirthDate = "10/12/1995";
CString strOneYearOldDate = AddIntervalToDate( strBirthDate, strInterval );
ASSERT( strOneYearOldDate == "11/12/1996" );
```

9.0 Provider Customization

In recent years, the ACIP has introduced into its recommendations specific areas within which providers may customize the administration of immunizations. For example, the age at which a particular dose of a vaccine is recommended may be represented by an age range, leaving it to individual practitioners to decide where, within the range, the dose should be given. Also, in the 1997 recommendations, three different sets of polio recommendations are described, with a statement that providers, and even parents, may choose among them.

These parameters may be contrasted with other aspects of the recommendations which, it may be presumed, are not candidates for customization. Providers should not, for example, elect to give oral polio vaccine to persons over 18 years of age and they should not continue to give DTP or DTaP vaccine past the age of seven.

From this perspective, simply giving providers the ability to define additional schedules for themselves would seem to be an inappropriate and unfocused way of implementing customization. Experience has shown that the process of setting the parameters of an evaluation algorithm to effect particular recommendations is very difficult and will probably remain a task for specialists.

The challenge for developers is to create mechanisms for customizing immunization schedules that will be easy for providers to use and which will focus on the areas of the recommendations that the ACIP has specifically marked for customization. This could be envisioned as a process, external to the parameter setting process, which would present to providers the age ranges defined for particular vaccine doses, allowing them to set a point within each range where they would prefer to administer the immunization. For this purpose, Valid Recommended Age Range has been included as a specifiable parameter in the data set shown in section 6.1. These parameters are not involved in the actual evaluation process -- they are used in the customization process.

Also, where the ACIP has specified different sets of recommendations for the same series, as with polio, providers could be presented with a simple choice of the alternatives for their selection. Each of the sets of recommendations would be specified in the regular series definition and the evaluation algorithm would dynamically select for use the one chosen by the provider. This choice could also be stored in patient records so that the selection of a particular alternate series could be specific to each patient.

10.0 Implementation

The implementation of an algorithm as complex as the one described presents several challenges to developers. Of paramount importance is speed of execution: the algorithm must be fast enough to be used as a real-time guide to current immunization needs in providers' offices as well as in immunization follow-up processes, in which hundreds or thousands of individual histories must be evaluated.

Careful consideration must be given to data structure design and execution efficiency. The choice of a programming platform is also important -- it is entirely possible that some programming environments may not be robust enough or powerful enough to support this mechanism adequately.

Some of the more salient implementation issues will now be discussed, using an actual implementation as an example: the CDC Model Immunization Information System. This is a Windows-hosted application, written completely in C++ for the Windows NT operating system, and using Microsoft's SQL Server as the database management system.

Note: there is no intention here to imply that the application or the computing platform represent an ideal implementation of the process; they will simply be used to illustrate the issues involved.

The first issue to address is the database design for the Schedule Definition Parameters. These parameters were described above as four data tables, linked by common fields. However, if the database structure followed this abstract description, retrieval of a definition would require a series of table joins, a relatively slow and cumbersome process. In the Model System, therefore, it was
decided to structure the data set as logical row types in a single table, keyed to the Schedule Definition Name. This way, an entire definition (which might involve 50-100 rows) could be retrieved by one SQL select statement with no joins.

Also, to simplify the structure, the Vaccine Dose Definition table was integrated into the Series Vaccine Definition table, as follows: each of the dose-specific parameters, like recommended interval and minimum age, is represented in the Vaccine Definition table as an array of numbers, contained in a string data type. This is a feasible approach since the number of doses in a series (and, hence, the number of array elements) is typically small, on the order of 5-10.

Note that, to support the single-table design, the columns for all the tables are included in all rows retrieved, regardless of whether they represent Schedule, Series or Vaccine Definitions. Columns that are not relevant to a logical row type are left null or blank; variable character strings are used to conserve data storage.

The second issue relates to the way a Schedule Definition is represented in memory. For execution efficiency, it is imperative that an entire Schedule Definition, as well as the individual's immunization and contraindication histories, be retrieved and maintained in memory while the evaluation process is carried out. In the Model System, each Series in a Definition is retrieved and stored in a structure, actually, a C++ object. As they are created, pointers to these objects are stored in an array. The algorithm, therefore, accesses the parameters for each series by traversing this pointer array. Within each Series object, Vaccine parameters are maintained in a set of string arrays, in which each element pertains to a given vaccine in the Series list. Note that, in the case of Vaccine Dose parameters, each string in a specified array represents a table of dose-specific parameters for a given vaccine.

The representation of a Schedule Definition in memory is a relatively straightforward affair in an object-oriented, or other, environment in which structures and pointers to structures can be created and manipulated. The ability to create dynamically sizable arrays is important, too. Programming environments without these features force the developer to exercise ingenuity in setting up the process.

The third issue involves the design of a user process for creating and maintaining Schedule Definitions. The process should allow easy access to, and manipulation of, any of the parameters in a Definition. Graphical User Interfaces, like Windows, provide a convenient environment for this purpose and there are many ways in which the process could be structured, so it will not be explored in great detail.

11.0 Validation of Immunization Evaluation Algorithms

An immunization evaluation algorithm should produce the right output (recommend the right vaccines) for all possible inputs (immunization and contraindication histories). While it is not possible to enumerate all combinations of input, it should be feasible to create a complement of representative cases that would demonstrate the validity of a particular implementation.

The National Immunization Program has created such a complement of test immunization histories (including contraindication histories) with corresponding recommendations that have been validated against the ACIP recommendations. These test cases are available in an ASCII text file that can be used by developers to evaluate their own algorithms. A methodology for using the test cases are given in the next section.

If an algorithm produces results that disagree with the outputs shown, it is presumed that the algorithm is incorrect. If the algorithm is completely embedded in program source code, then the code would have to be changed to correct the problem. In the case of a parameterized algorithm, the problem may lie either in the code or in the parameter settings.

It is, of course, possible that a test case may be wrong. Disputes will be arbitrated by National Immunization Program staff who work with the ACIP recommendations.

Feedback is invited from developers who use this mechanism to validate their evaluation algorithms.

12.0 A Methodology for Evaluating an Algorithm Using the NIP Test Cases

In general, the approach to validating an immunization evaluation algorithm involves writing a computer program that will do the following:

1. Read each test case in the ASCII file

2. Execute the algorithm, using the relevant history data as input parameters

3. Compare the output of the algorithm to the recommendations in the test case data

12.1 ASCII File Data Structure

The ASCII file represents the test cases as a set of data rows, each row corresponding to one of three record types:

1. Immunization (Record Type = I), denoting a single immunization
2. Contraindication (Record Type = C), denoting a single contraindication
3. Recommendation (Record Type = R), denoting a single recommendation

Each test case is comprised of one or more data rows. Every case has at least one Recommendation row and 0, 1 or more Immunization and/or Contraindication rows. For example, the first test case consists of one Recommendation row and no Immunization or Contraindication rows. This case gives the recommendation for a child who has had no immunizations and who has no contraindications. All the data rows for a given case are linked by a *Case ID* number.

The field layout for the data rows is:

Field Name	Data Type	Length	Position	Description
Record Type	Alpha	1	1	Values: R, I, or C
Case ID	Numeric	5	2	Sequential Number
Birth Date	Numeric	8	7	YYYYMMDD
Evaluation Date	Numeric	8	15	YYYYMMDD
Age at Evaluation	Alpha	16	23	y = year, m = month, d = day
Vaccine Type	Alpha	16	39	meaning depends on record type*
Event Date	Numeric	8	55	meaning depends on record type**
Recommendation Code	Numeric	1	63	valid values: 1-4***
Immunizaton Series	Alpha	32	64	Series Being Evaluated

* Record Type I: represents immunization administered
Record Type C: represents vaccine that is contraindicated
Record Type R: represents vaccine that is recommended

** Record Type I: represents date that immunization was administered
Record Type C: represents date that contraindication expires
Record Type R: always blank

*** 1 = Complete 2 = Up To Date 3 = Recommendation 4 = No Recommendation

12.2 Vaccine Type Name Translation

The vaccine type names used in the test cases are obtained from the HL7 Immunization Data Standard (V2.3). Vaccine names used in Immunization Data Systems may be different from these. Computer validation programs, therefore, will have to create a translation table, mapping the vaccine names, used by the test cases, to those that will be recognized by the algorithm being tested.

HL7 Vaccine Table

Value	Description	Vaccine Name

54	Adenovirus Type 4	Adenovirus Type 4, live, oral
55	Adenovirus Type 7	Adenovirus Type 7, live, oral
24	Anthrax	Anthrax
19	BCG	Bacillus of Calmette & Guerin
27	Botulinum antitoxin	Botulinum antitoxin
26	Cholera	Cholera
29	CMVIG	Cytomegalovirus immune globulin, intravenous
56	Dengue Fever	Dengue Fever
12	Diphtheria antitoxin	Diphtheria antitoxin
28	DT (Pediatric)	Diphtheria & tetanus toxoids
20	DTaP	Diphtheria-tetanus-acellular pertussis
50	DTaP-Hib	DTaP-Haemophilus influenzae type b conjugate
01	DTP	Diphtheria-tetanus-pertussis
22	DTP-Hib	DTP-Haemophilus influenzae type b conjugate
57	Hantavirus	Hantavirus
30	HBIG	Hepatitis B immune globulin
31	Hep A--(Pediatric)	Hepatitis A
52	Hep A--(Adult)	Hepatitis A
45	Hep B--other or unspecified	Hepatitis B--other or unspecified
08	Hep B--adolescent or pediatric	Hepatitis B--adolescent or pediatric
42	Hep B--adolescent/high risk infant	Hepatitis B--adolescent/high risk infant
43	Hep B--adult	Hepatitis B--adult
44	Hep B--dialysis	Hepatitis B--dialysis
58	Hepatitis C	Hepatitis C
59	Hepatitis E	Hepatitis E
60	Herpes Simplex 2	Herpes Simplex 2
17	Hib--unspecified	Haemophilus influenzae type b conjugate-unspecified
46	Hib--PRP-D	Haemophilus influenzae type b conjugate--PRP-D
47	Hib--HbOC	Haemophilus influenzae type b conjugate--HbOC
48	Hib--PRP-T	Haemophilus influenzae type b conjugate--PRP-T
49	Hib--PRP-OMP	Haemophilus influenzae type b conjugate--PRP-OMP
51	Hib-Hep B	Haemophilus influenzae type b conjugate-Hep B
61	HIV	Human Immunodeficiency Virus
62	Human Papilloma Virus	Human Papilloma Virus
14	IG	Immune globulin
15	Influenza--split (incl. purified surface antigen)	Influenza--split (incl. purified surface antigen)
16	Influenza--whole	Influenza--whole
10	IPV	Poliovirus vaccine, inactivated
39	Japanese encephalitis	Japanese encephalitis
63	Junin Virus	Junin Virus
64	Leishmaniassis	Leishmaniassis
65	Leprosy	Leprosy
66	Lyme Disease	Lyme Disease
03	MMR	Measles-mumps-rubella
04	M/R	Measles & rubella
67	Malaria	Malaria
05	Measles	Measles
68	Melanoma	Melanoma
32	Meningococcal	Meningococcal
07	Mumps	Mumps

69	Parainfluenza-3 Virus	Parainfluenza-3 Virus
11	Pertussis	Pertussis
23	Plague	Plague
33	Pneumococcal	Pneumococcal
02	OPV	Poliovirus vaccine, oral
70	Q Fever	Q Fever
18	Rabies--intramuscular injection	Rabies--intramuscular injection
40	Rabies--intradermal injection	Rabies--intradermal injection
72	Rheumatic Fever	Rheumatic Fever
73	Rift Valley Fever	Rift Valley Fever
34	RIG	Rabies immune globulin
74	Rotavirus	Rotavirus
71	RSV-IGIV	Respiratory Syncytial Virus Immune Globulin, intravenous
06	Rubella	Rubella
38	Rubella/Mumps	Rubella & Mumps
75	Smallpox	Smallpox
76	Staphylococcus Bacterio Lysate	Staphylococcus Bacterio Lysate
09	Td (Adult)	Tetanus-diphtheria
35	Tetanus toxoid	Tetanus toxoid
77	Tick-borne Encephalitis	Tick-borne Encephalitis
13	TIG	Tetanus immune globulin
78	Tularemia	Tularemia
25	Typhoid--oral	Typhoid--oral
41	Typhoid--parenteral	Typhoid--parenteral
53	Typhoid--parenteral, AKD (U.S. military)	Typhoid--parenteral, acetone-inactivated (U.S. military)
79	Vaccinia Immune Globulin	Vaccinia Immune Globulin
21	Varicella	Varicella
81	VEE--inactivated	Venezuelan Equine Encephalitis--inactivated
80	VEE--live, attenuated	Venezuelan Equine Encephalitis--live, attenuated
36	VZIG	Varicella zoster immune globulin
37	Yellow fever	Yellow fever

12.3 Algorithm Validation Procedure

The computer program that will validate an algorithm could be structured as a procedural loop in which each test case is read in turn and submitted to the algorithm for evaluation and comparison to the standard recommendations. The structure of such a procedure is shown below:

```
while( TRUE )
{
        ReadNextCase();
        if ( EOF )
            break;
        TranslateVaccineTypeNames();
        ExecuteEvaluationAlgorithm( TestCaseData );
        CompareRecommendations();
}
```

12.4 Evaluation of Results

Each test case evaluates one immunization series only and it provides recommendations for all vaccines that would be recommended for that series, with the preferred recommendation appearing first in the list. For example, in the test case for the DTP/DT/DTaP/Td series, involving a child who is two months old and has no immunizations, the recommendations are:

DTP/PRP-T
DTP-HbOC
DTP
DT ped

In contrast, the algorithm being validated may make recommendations for all series on each execution. This suggests a simple strategy for scoring the algorithm's performance:

1. if the algorithm recommends the vaccine that appears *first* on the standard list, add 1 to an A-Tally.

2. if the algorithm recommends *any* vaccine on the standard list, add 1 to a B-Tally.

3. if the standard recommendation is *"Up To Date"* or *"No Recommendation"*, and the algorithm recommends a vaccine that appears on the list for the *next* test case for the same series, add 1 to a C-Tally.

4. if the standard recommendation is "*Complete"*, and the algorithm recommends a vaccine that appears on the list for the *previous* test case for the same series, add 1 to the C-Tally.

At the end of the run, subtract the C-Tally total from both the A- and B-Tally totals. Then, express the resulting totals as percentages of the total number of test cases in the case complement. The higher the percentage on either of the resulting tallies, the better is the agreement with the standard. The A-Tally, of course, represents a stricter test. Perfect agreement with the standard would be indicated by 100% scores on both the A- and B-Tallies.

Attachments

1. Algorithm Parameters Data File (parmdata.txt)
2. Algorithm Parameters Text File (parmtext.txt)
3. Test Cases Data File (testdata.txt)
4. Test Cases Text File (testtext.txt)